



Intro to Computer Science

Summer 2020

August 4th to August 20th, online

Data Structures- Dictionaries

Adopted from Stanford Uni's [CS106B course slides](#) by Kate Jost and Sarah Johnson; by and [Code as Poetry](#) by Perik and Sahar; Koca Uni's [Comp123 course](#) by Ayca Tuzmen

Today's questions

How can I organize my data so it's
easier to use?

How can I organize my data so it's easier to use?

Think/Share:

Store names of habitat animals and their corresponding diet



elephant



bear



otter



platypus



clams



grass



shrimp



berries

Task - Relating data with each other

```
['elephant', 'bear', 'otter', 'platypus']  
['grass', 'berries', 'clams', 'shrimp']
```

Task - Relating data with each other

```
['elephant', 'bear', 'otter', 'platypus']
```

```
['grass', 'berries', 'clams', 'shrimp']
```

 These pieces of
information are linked!

Task - Relating data with each other

```
['elephant', 'bear', 'otter', 'platypus']
```

```
['grass', 'berries', 'clams', 'shrimp']
```



These pieces of
information are linked!

Can we store them so they're
associated with each other?

Dictionaries!

Definition

Dictionary

A container data type that maps “keys” to their associated “values”.

Anatomy of a Dictionary

```
name_of_dic = {}
```

```
name_of_dic = {'elephant': 'grass', 'bear': 'berries',  
'otter': 'clams', 'platypus': 'shrimp'}
```

Anatomy of a Dictionary

```
name_of_dic = {'elephant': 'grass', 'bear': 'berries',  
'otter': 'clams', 'platypus': 'shrimp'}
```



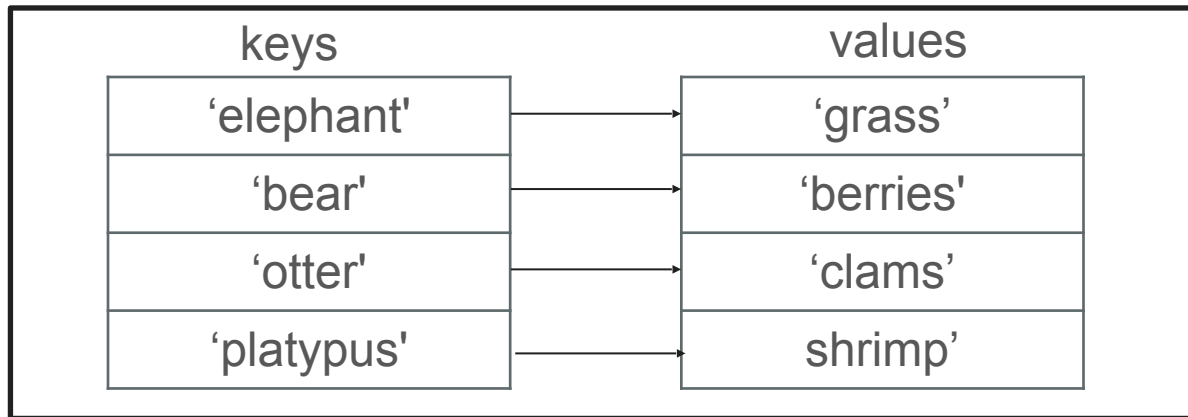
This is a dictionary literal

Anatomy of a Dictionary

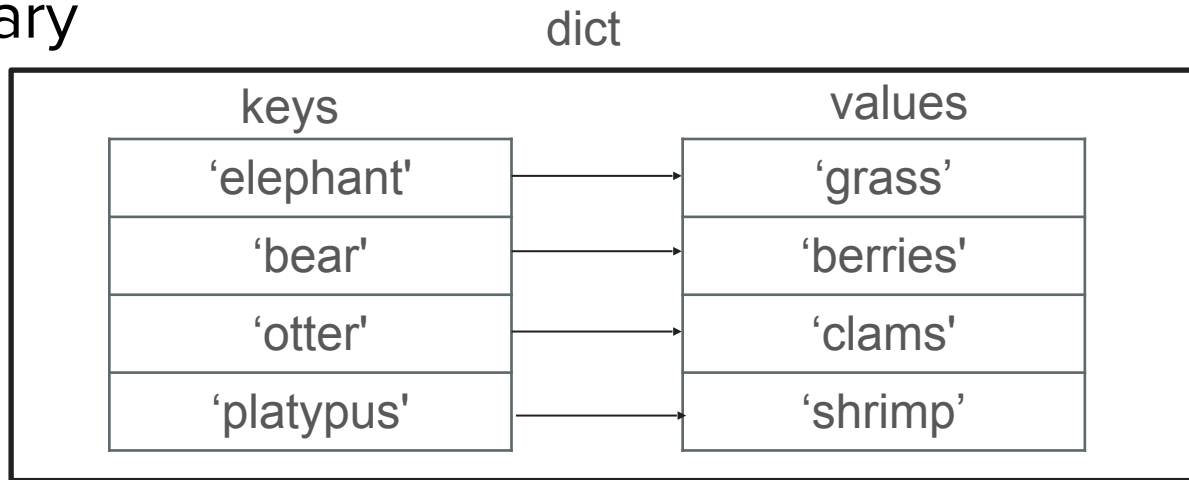
```
name_of_dic = {'elephant': 'grass', 'bear': 'berries',  
'otter': 'clams', 'platypus': 'shrimp'}
```



It is easier to visualize it this way:



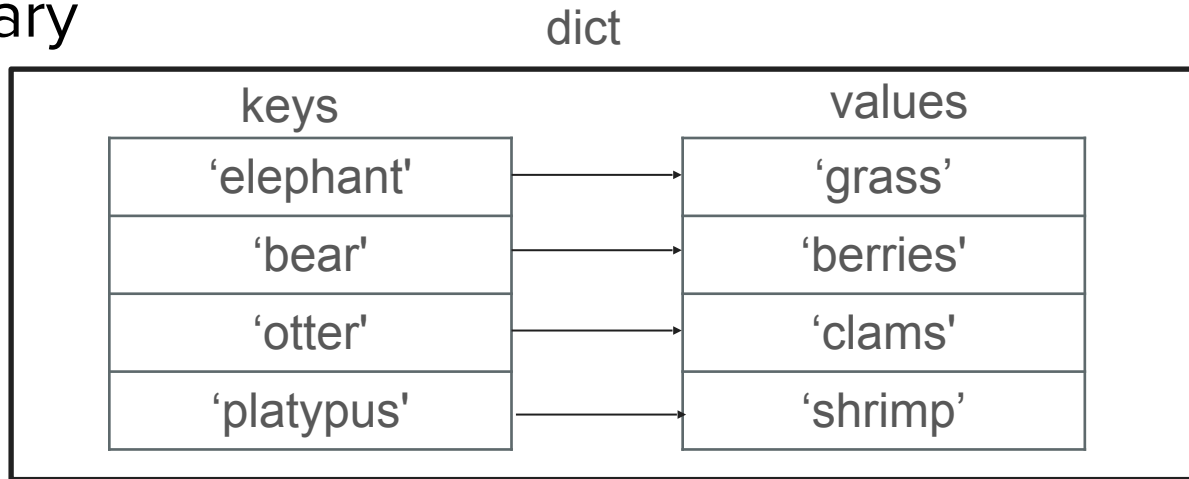
Anatomy of a Dictionary



Each key can store
one value

Anatomy of a Dictionary

```
>>> d[ 'elephant' ]
```



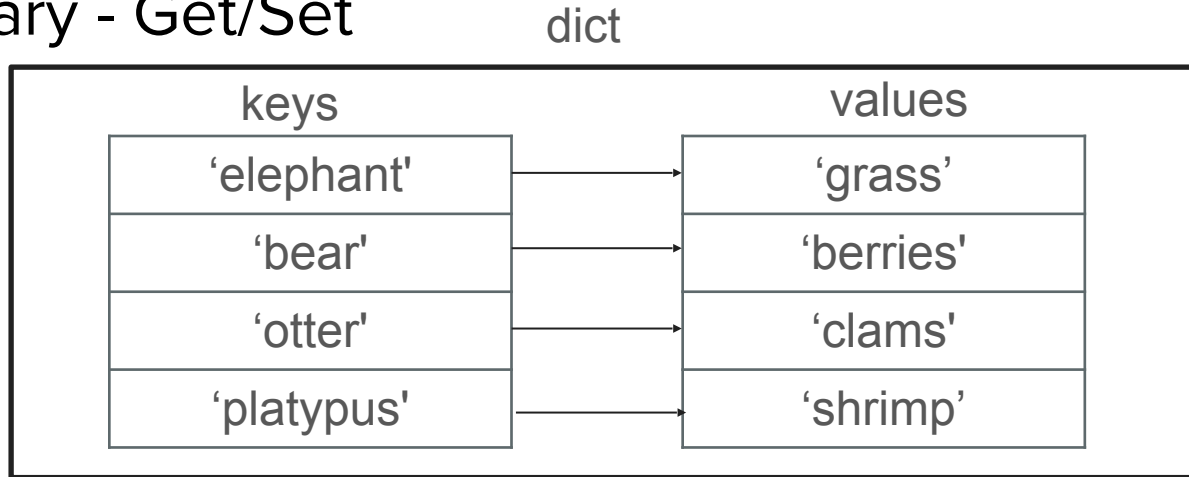
Each key can store
one value

Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```



This operation
is called “get”

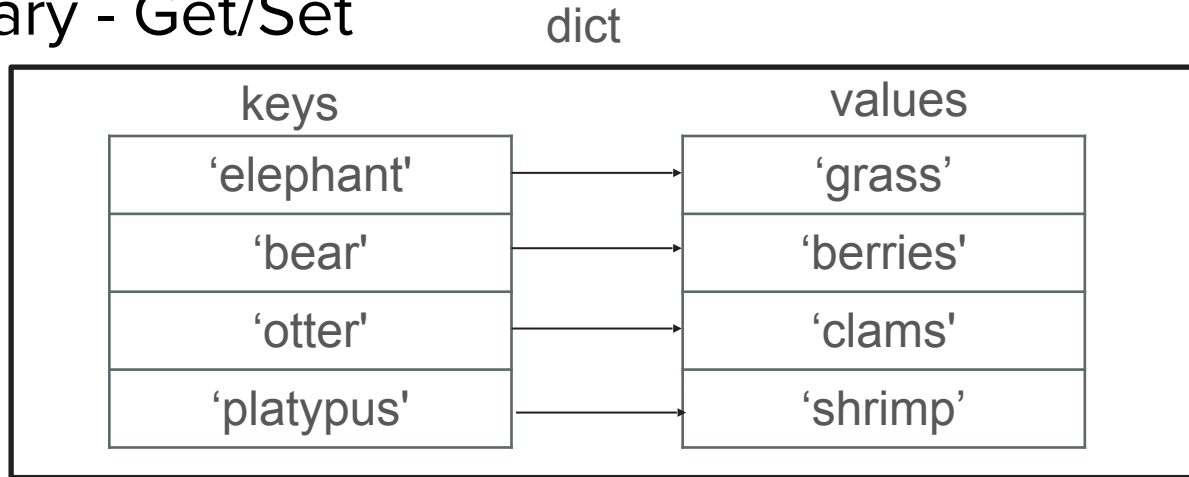


Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```



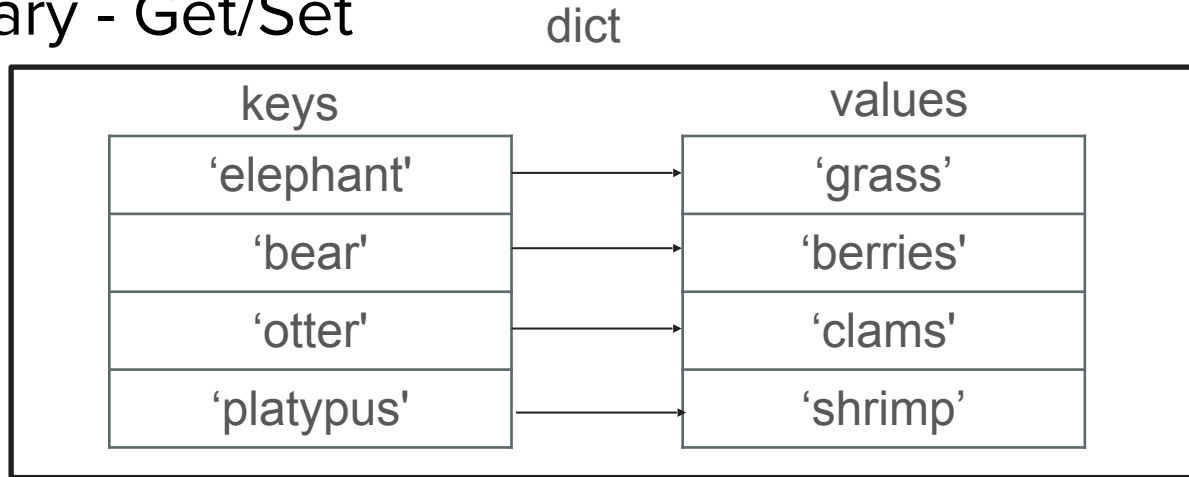
This operation
is called “get”



Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```

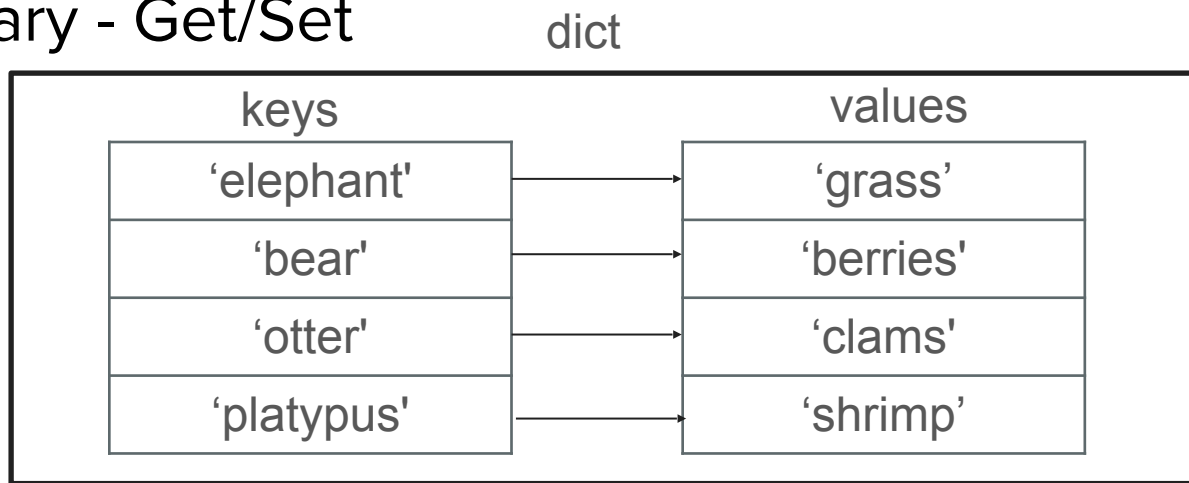
'grass'



Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```

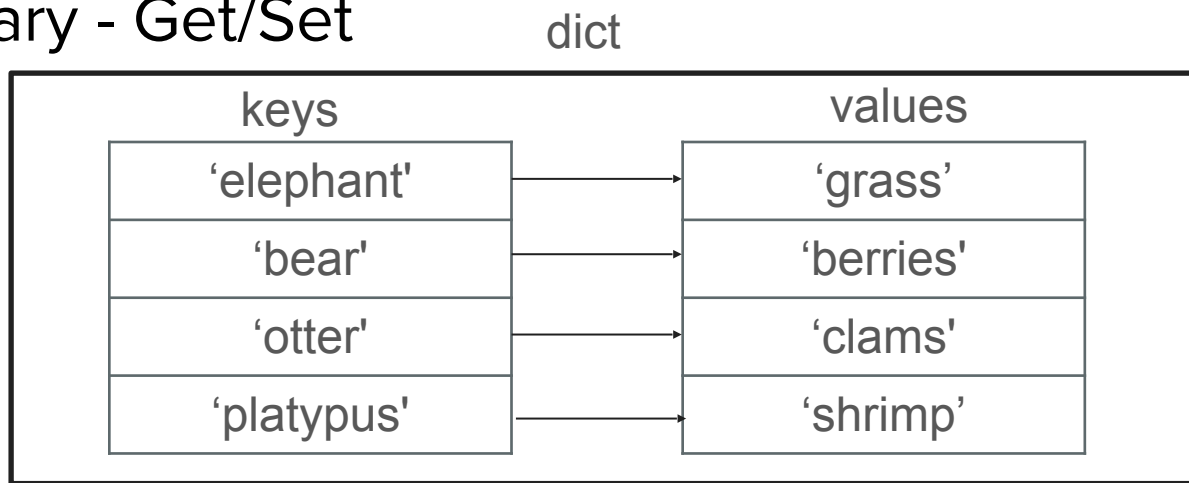
'grass'



Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```

'grass'



```
>>> d['elephant'] = 'leaves'
```

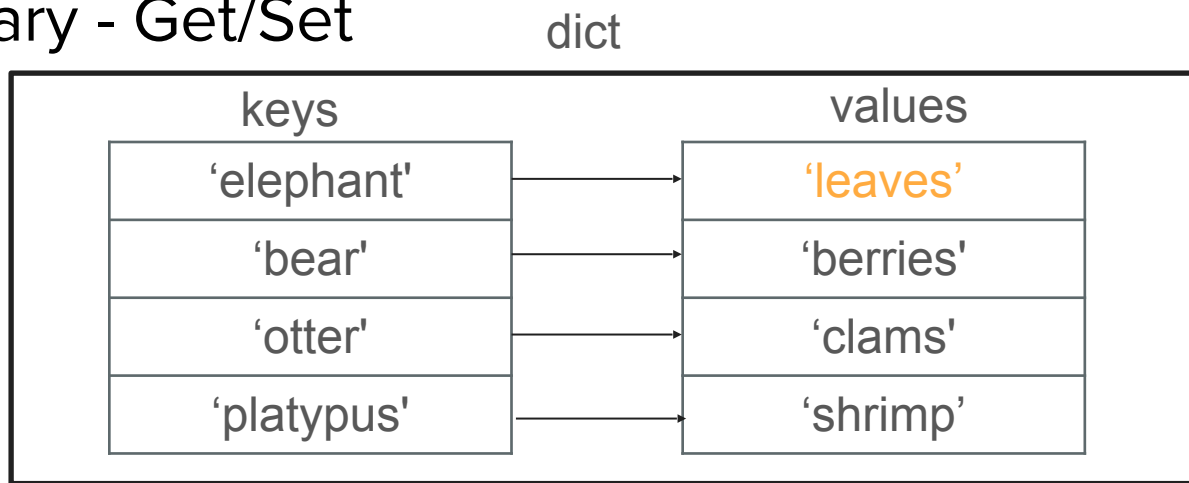


This operation
is called “set”

Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```

`'grass'`



```
>>> d['elephant'] = 'leaves'
```

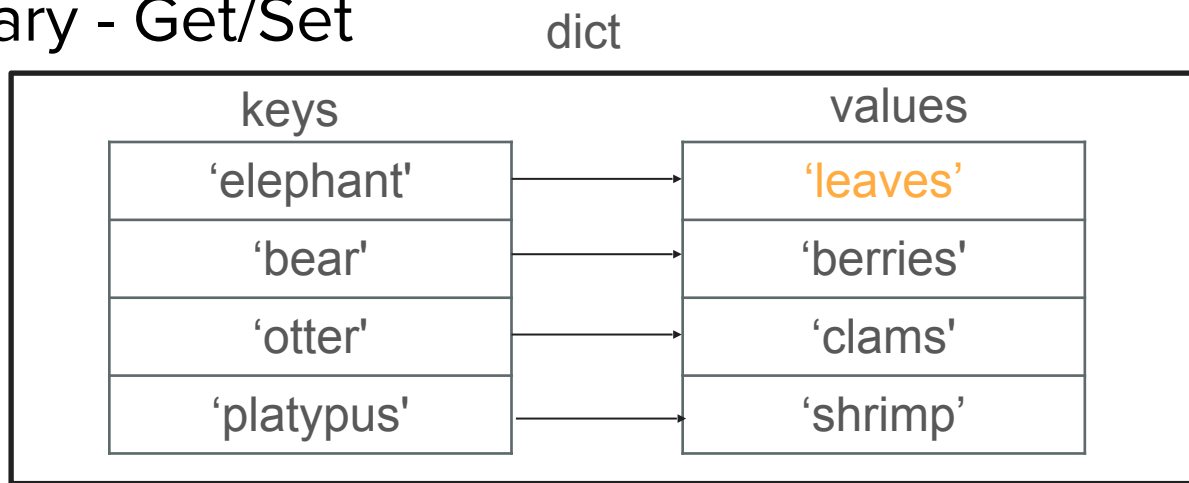


This operation
is called “set”

Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```

```
'grass'
```



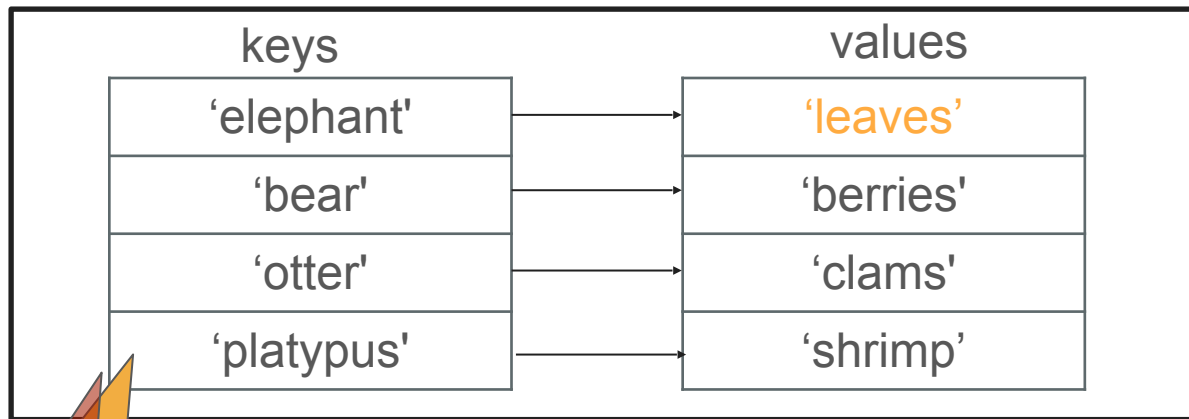
```
>>> d['elephant'] = 'leaves'
```

```
>>> d['cat']
```

Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```

'grass'



```
>>> d['elephant']
```

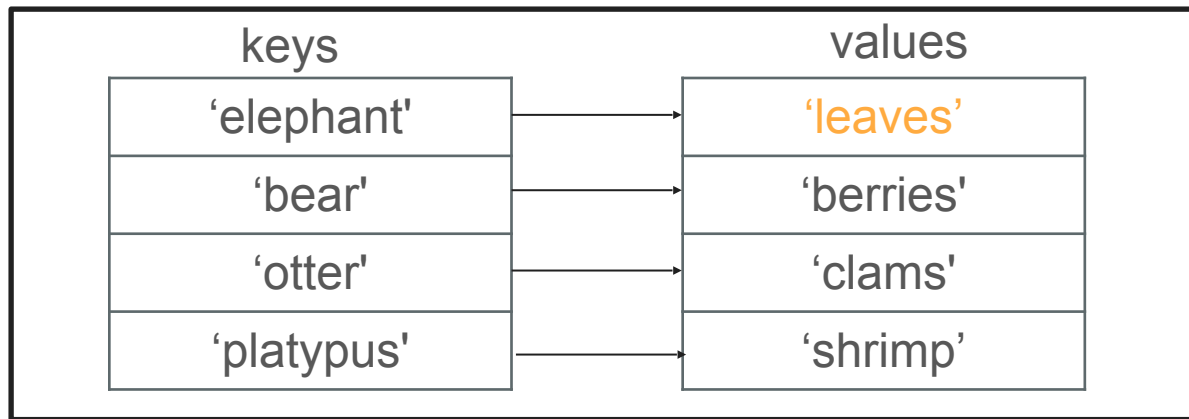
```
>>> d['cat']
```

KeyError

Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```

`'grass'`



```
>>> d['elephant'] = 'leaves'
```

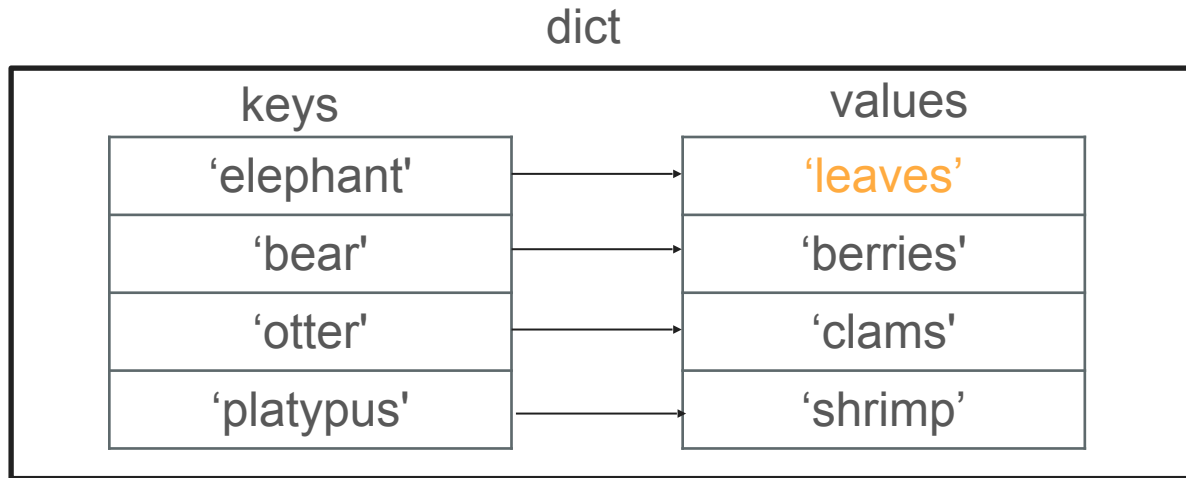
```
>>> d['cat']
```



“get” errors if the key
is not in the dict

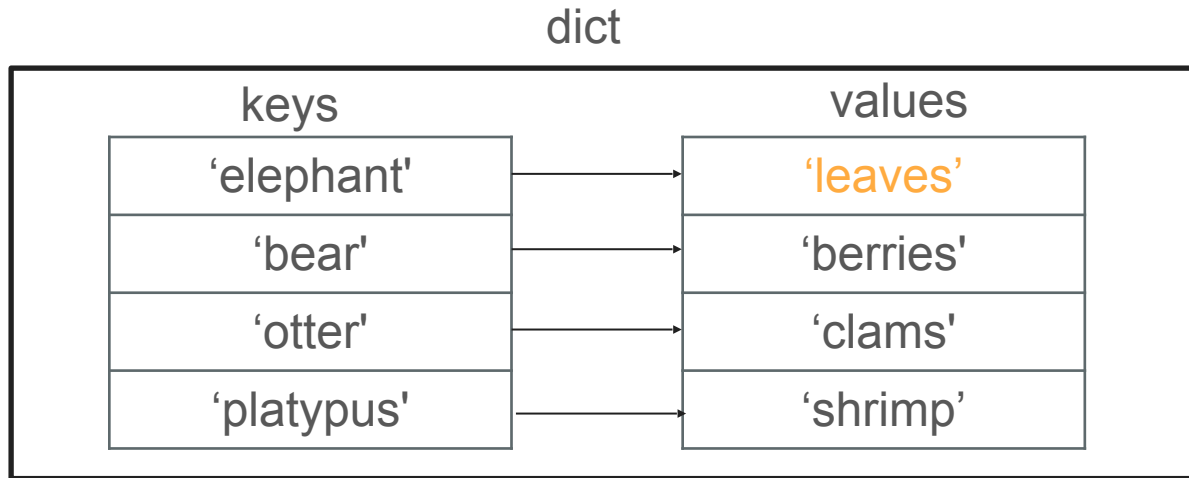
Dictionary - in

```
>>> 'elephant' in d
```



Dictionary - in

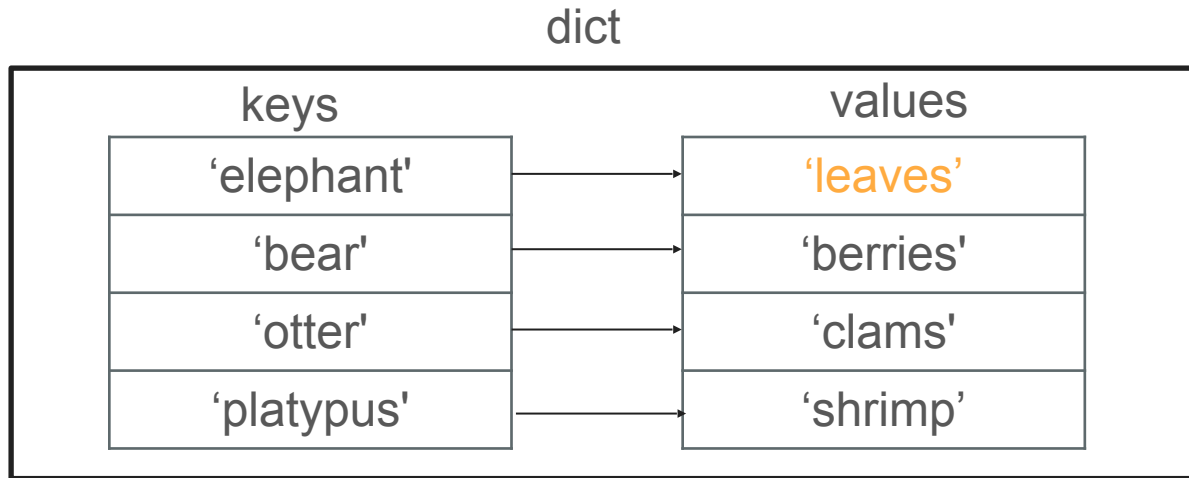
```
>>> 'elephant' in d  
True
```



Dictionary - in

```
>>> 'elephant' in d  
True
```

```
>>> 'cat' not in d  
True
```



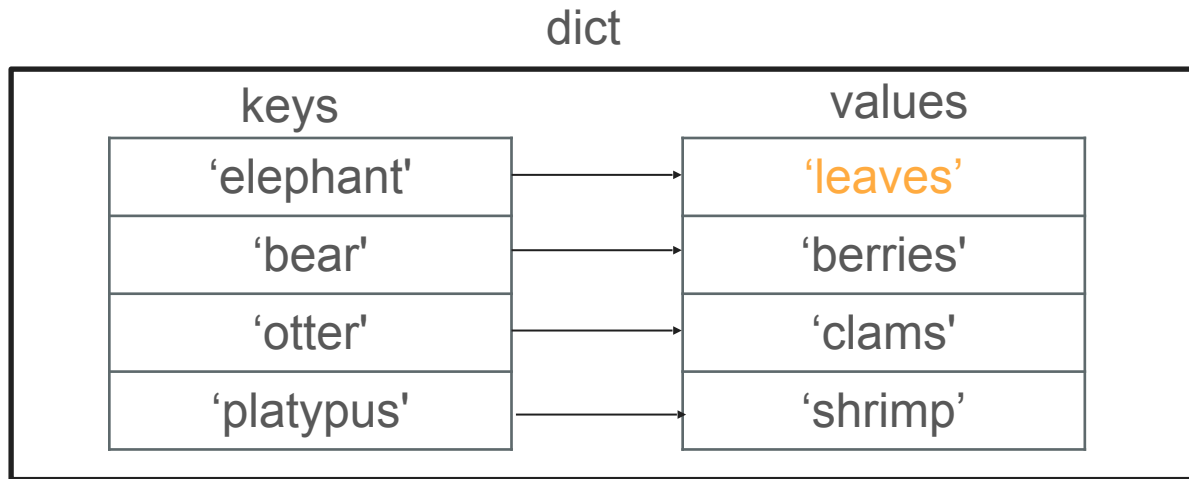
Dictionary - in

```
>>> 'elephant' in d
```

```
True
```

```
>>> 'cat' not in d
```

```
True
```



Common pattern: Check if key is present.
If it is, do something. If it isn't, do
something else.

Building a dictionary

```
>>> d = {}
```

Building a dictionary

```
>>> d = {}
```



Create an empty
dictionary

Building a dictionary

```
>>> d = {}
```

```
>>> d['elephant'] = 'grass'
```

Building a dictionary

```
>>> d = {}
```

```
>>> d['elephant'] = 'grass'
```



We can add keys using “set”

Building a dictionary

```
>>> d = {}
```

```
>>> d['elephant'] = 'grass'
```



```
>>> d
```

We can add keys using “set”

Building a dictionary

```
>>> d = {}
```

```
>>> d['elephant'] = 'grass'
```



```
>>> d
```

```
{ 'elephant': 'grass' }
```

We can add keys using “set”

Building a dictionary

```
>>> d = { 'elephant' : 'grass' }
```

Types of Dictionaries

- So far, we've seen dictionaries mapping from strings to ints
 - This is not the only type of dictionary!
 - You can map from string/int/float to string/int/float...

Think/Share:

Store names of CS lecturers and their ages

Accessing a Dictionary's Keys

```
>>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

Building a dictionary

```
>>> d = { 'Ayca' : 34 }
```

```
>>> d[ 'Ayca' ] += 2
```

Building a dictionary

```
>>> d = { 'Ayca' : 34 }
```

```
>>> d[ 'Ayca' ] += 2
```



we can get/set on the same line!
(same as `d['Ayca'] = d['Ayca'] + 2`)

Building a dictionary

```
>>> d = { 'Ayca' : 34 }
```

```
>>> d[ 'Ayca' ] += 2
```

```
>>> d[ 'Ayca' ]  
{ 'Ayca' : 36 }
```



we can get/set on the same line!
(same as `d['Ayca'] = d['Ayca'] + 2`)

Accessing a Dictionary's Keys

```
>>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30,  
         'Chris' = 29' }
```

```
>>> d.keys()
```


Accessing a Dictionary's Keys

```
>>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30,  
         'Chris' = 29' }
```

```
>>> d.keys()
```

```
dict_keys(['Ayca', 'Nick', 'Ondrej', 'Chris'])
```




Iterable collection of all the keys.
Iterable means it can be used in foreach

Accessing a Dictionary's Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> list(d.keys())  
[ 'Ayca' , 'Nick' , 'Ondrej' , Chris ]
```



we are using list() to convert
d.keys() into a list

Accessing a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

Accessing a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> list(d.values())
```

Accessing a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> list(d.values())
```



we are using `list()` to convert
`d.values()` into a list

Accessing a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> list(d.values())
```

```
[34, 28, 30, 29]
```



we are using list() to convert
d.values() into a list

Looping over a Dictionary's Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

Looping over a Dictionary's Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name in d.keys():
```


Looping over a Dictionary's Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name in d.keys():  
...     print(name)
```

Looping over a Dictionary's Keys

```
>>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name in d.keys():  
...     print(name)
```

Ayca

Nick

Ondrej

Chris

Looping over a Dictionary's Keys

```
>>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name in d.keys():  
...     print(name)
```

Ayca

Nick

Ondrej

Chris



we can use foreach on the
dictionary's keys!

Looping over a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

Looping over a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for age in d.values():
```

Looping over a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for age in d.values():  
...     print(age)
```

Looping over a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for age in d.values():  
...     print(age)
```

34

28

30

29

Looping over a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```


```
>>> for age in d.values():  
...     print(age)
```

34

28

30

29



we can use foreach on the
dictionary's values!

Looping over a Dictionary's Keys and Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

Looping over a Dictionary's Keys and Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name, age in d.items():
```

Looping over a Dictionary's Keys and Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name, age in d.items():
```




items() gives us
key, value pairs

Looping over a Dictionary's Keys and Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name, age in d.items():
```

```
...     print(name, 'is', age, 'years old.')
```



items() gives us
key, value pairs

Looping over a Dictionary's Keys and Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```


```
>>> for name, age in d.items():  
...     print(name, 'is', age, 'years old.')
```

Ayca is 34 years old.

Nick is 28 years old.

Ondrej is 30 years old.

Chris is 29 years old.



items() gives us
key, value pairs

Looping over a Dictionary's Keys and Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```


```
>>> for name, age in d.items():  
...     print(name, 'is', age, 'years old.')
```

Ayca is 34 years old.

Nick is 28 years old.

Ondrej is 30 years old.

Chris is 29 years old.

 print() will automatically concatenate args separated by commas!

Printing with sep=


```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name, age in d.items():  
...     print(name, age, sep=': ')
```

Printing with sep=

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name, age in d.items():  
...     print(name, age, sep=': ')
```

 sep is an optional
argument like end!

Printing with sep=

```
>>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name, age in d.items():  
...     print(name, age, sep=': ')
```

Ayca: 34

Nick: 28

Ondrej: 30

Chris: 29



sep is an optional
argument like end!

Printing with sep=

```
>>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name, age in d.items():  
...     print(name, age, sep=': ')
```

Ayca: 34

Nick: 28

Ondrej: 30

Chris: 29



the separating string will
be printed between the
arguments you pass into
print()

Printing with sep=

```
>>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> for name, age in d.items():  
...     print(name, age, sep=': ')
```

Ayca: 34

Nick: 28

Ondrej: 30

Chris: 29



the default is sep=': ' (insert space)

Getting a Sorted List of Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

Getting a Sorted List of Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> sorted(d.keys())
```

Getting a Sorted List of Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> sorted(d.keys())  
[ 'Ayca' , 'Chris' , 'Nick' , 'Ondrej' ]
```

Getting a Sorted List of Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> sorted(d.keys())
```

```
['Ayca', 'Chris', 'Nick', 'Ondrej']
```



sorted() returns a list in
alphabetical order!

Getting a Sorted List of Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> sorted(d.keys())  
[ 'Ayca' , 'Chris' , 'Nick' , 'Ondrej' ]  
>>> d
```


Getting a Sorted List of Keys

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> sorted(d.keys())
```

```
['Ayca', 'Chris', 'Nick', 'Ondrej']
```

```
>>> d
```

```
['Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' = 29']
```

Sorting a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

Sorting a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> sorted(d.values())
```

Sorting a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> sorted(d.values())  
[28, 29, 30, 34]
```

Sorting a Dictionary's Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> sorted(d.values())
```

```
[28, 29, 30, 34]
```



sorted() returns a list in
numerical order!

Retrieving Min/Max Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

Retrieving Min/Max Values


```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> min(d.values())
```

Retrieving Min/Max Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> min(d.values())
```




returns the smallest
element!

Retrieving Min/Max Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> min(d.values())
```

28




returns the smallest
element!

Retrieving Min/Max Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> min(d.values())  
28
```



returns the smallest
element!

```
>>> max(d.values())
```

Retrieving Min/Max Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> min(d.values())
```

28

```
>>> max(d.values())
```



returns the smallest
element!



returns the biggest
element!

Retrieving Min/Max Values

```
>> d = { 'Ayca' : 34, 'Nick' : 28, 'Ondrej' : 30, 'Chris' =  
29' }
```

```
>>> min(d.values())
```

28

```
>>> max(d.values())
```

34



returns the smallest
element!



returns the biggest
element!

What's next?

Think/Share:

Implement a phone book using
dictionaries

Nested Data Structures

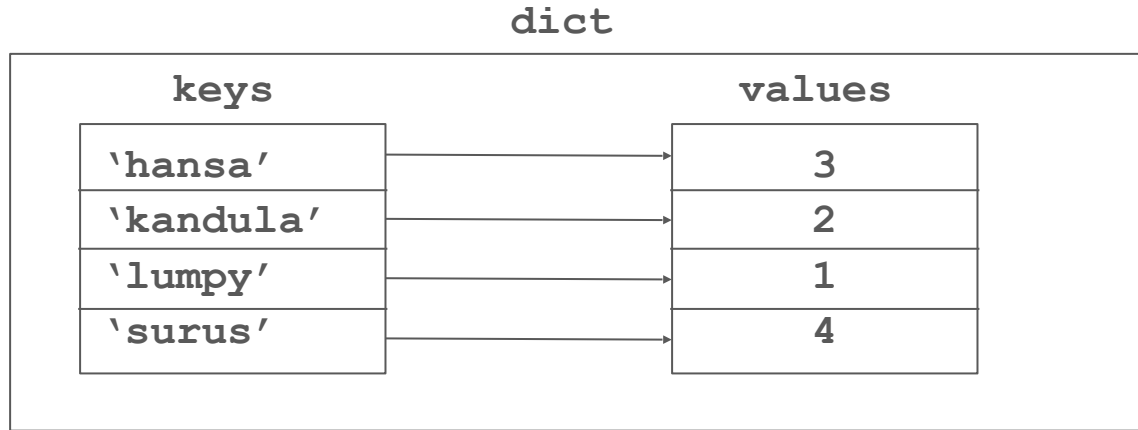
- We can nest data structures!
 - Lists in lists
 - *grid/game board*
 - Lists in dicts
 - *animals to feeding times*
 - Dicts in dicts
 - *your phone's contact book*
 - ... and so on!

Think/Share:

Make a dictionary of habitat animals and the number of times each animal has been fed.

Animal – Feedings Dictionary

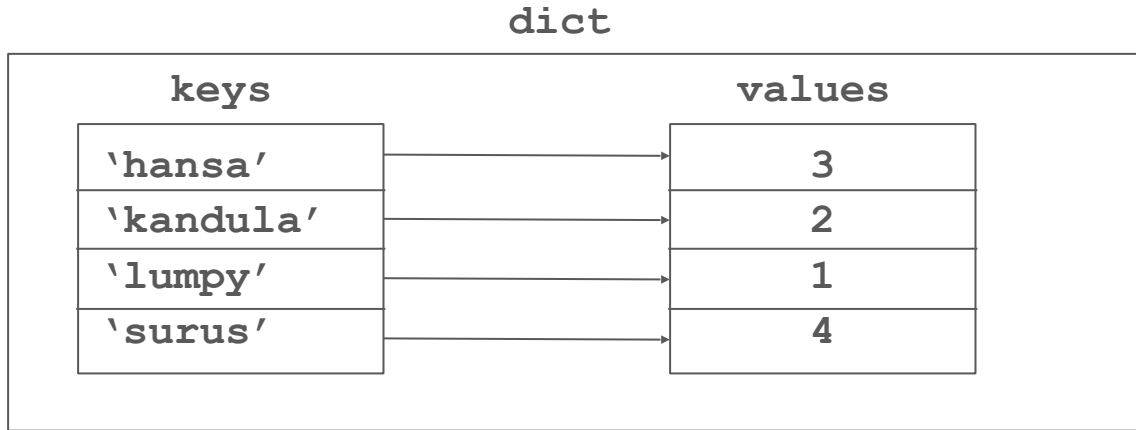
- animal name → number of feedings
- string → int



Recall: Animal – Feedings Dictionary

- animal name →
number of feedings
- string → int

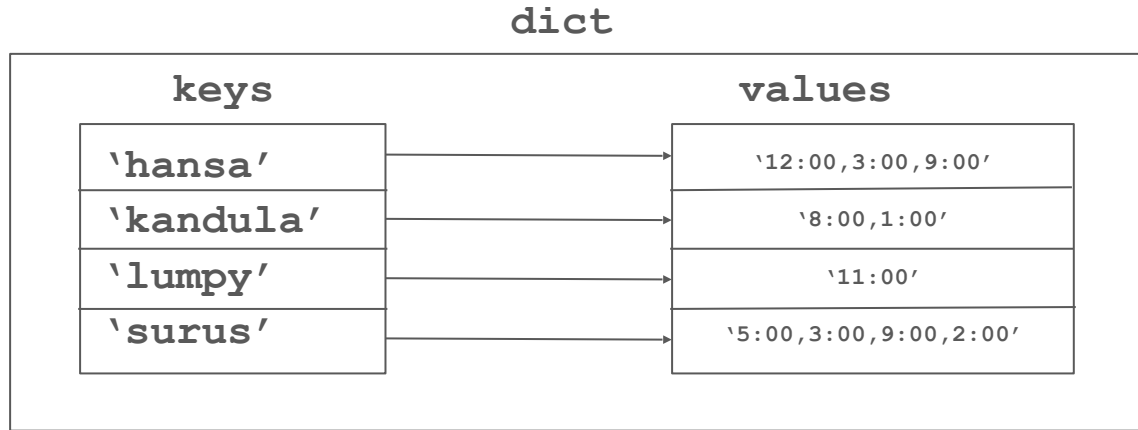
What if we wanted
to store the **times**
that the animals
were fed?



Attempt #1: Animal – Feeding Times Dictionary

- animal name → **feeding times**
- string → **string**

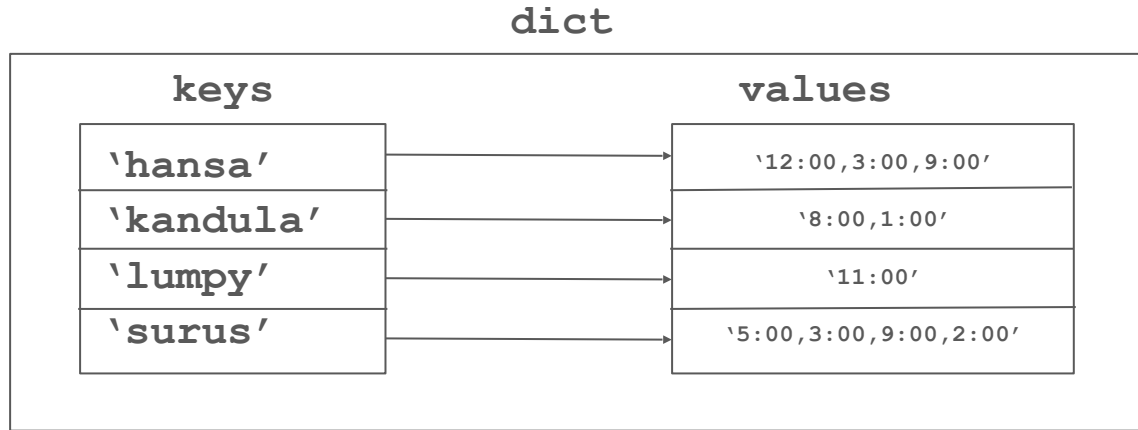
What if we wanted to store the **times** that the animals were fed?



Attempt #1: Animal – Feeding Times Dictionary

- animal name → **feeding times**
- string → **string**

What if we wanted to store the **times** that the animals were fed?

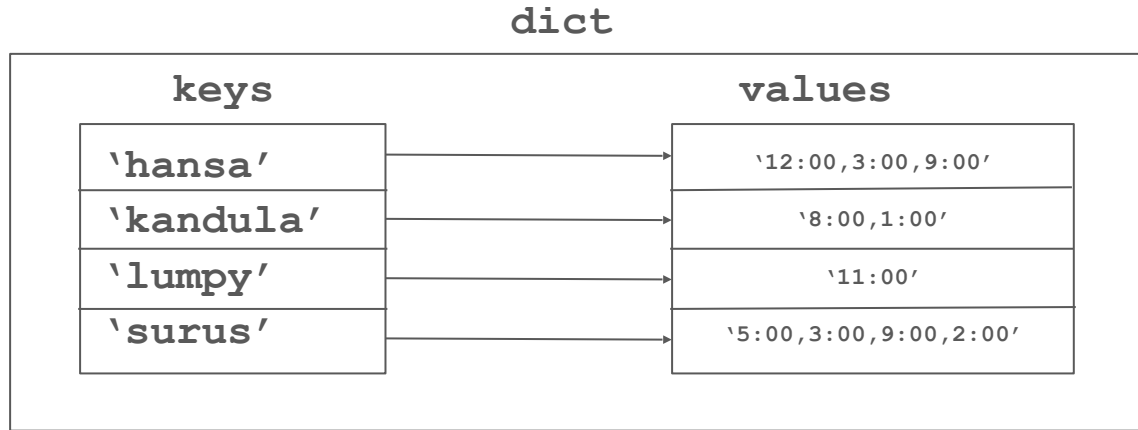


✗ Times are not easily accessible!

Attempt #1: Animal – Feeding Times Dictionary

- animal name → **feeding times**
- string → **string**

What if we wanted to store the **times** that the animals were fed?



But those times look like a data type we know of.....

Attempt #2: Animal – Feeding Times Dictionary

- animal name → **feeding times**
- string → **list[string]**

What if we wanted to store the **times** that the animals were fed?

